NIST Reference Platform and API Implementation

Robert Snelick

National Institute of Standards & Technology (NIST)

rsnelick@nist.gov

www.dase.nist.gov

Support: ITL & NIST ATP (Advanced Technology Program)

NIST Reference Platform

- Open API Reference Implementation
- Sample DASE Applications
- API Unit Tests
- Tools
- STB Platforms
  - NIST STB Simulation
  - Real-time systems

Why a Reference Implementation?

- Proof of Concept
  - detect inconsistencies and holes in API
  - “benchmark” implementation
- Conformance Testing
  - test against an implementation
- Application Development & Testing
  - accelerate application development process

Benefits of a NIST RI

- Neutral, 3rd party
  - no bias
  - no preconceived notions
- Prototype Source Code
  - public domain
  - basis for an implementation
  - component placeholders (modular)

Reference Platform Scope (I)

- DASE Reference Implementation
  - What we are Doing:
    - DASE Java APIs (java.tv, atsc, davic, havi)
    - Application Management
  - What we are NOT Doing (potential follow on):
    - Presentation Engine

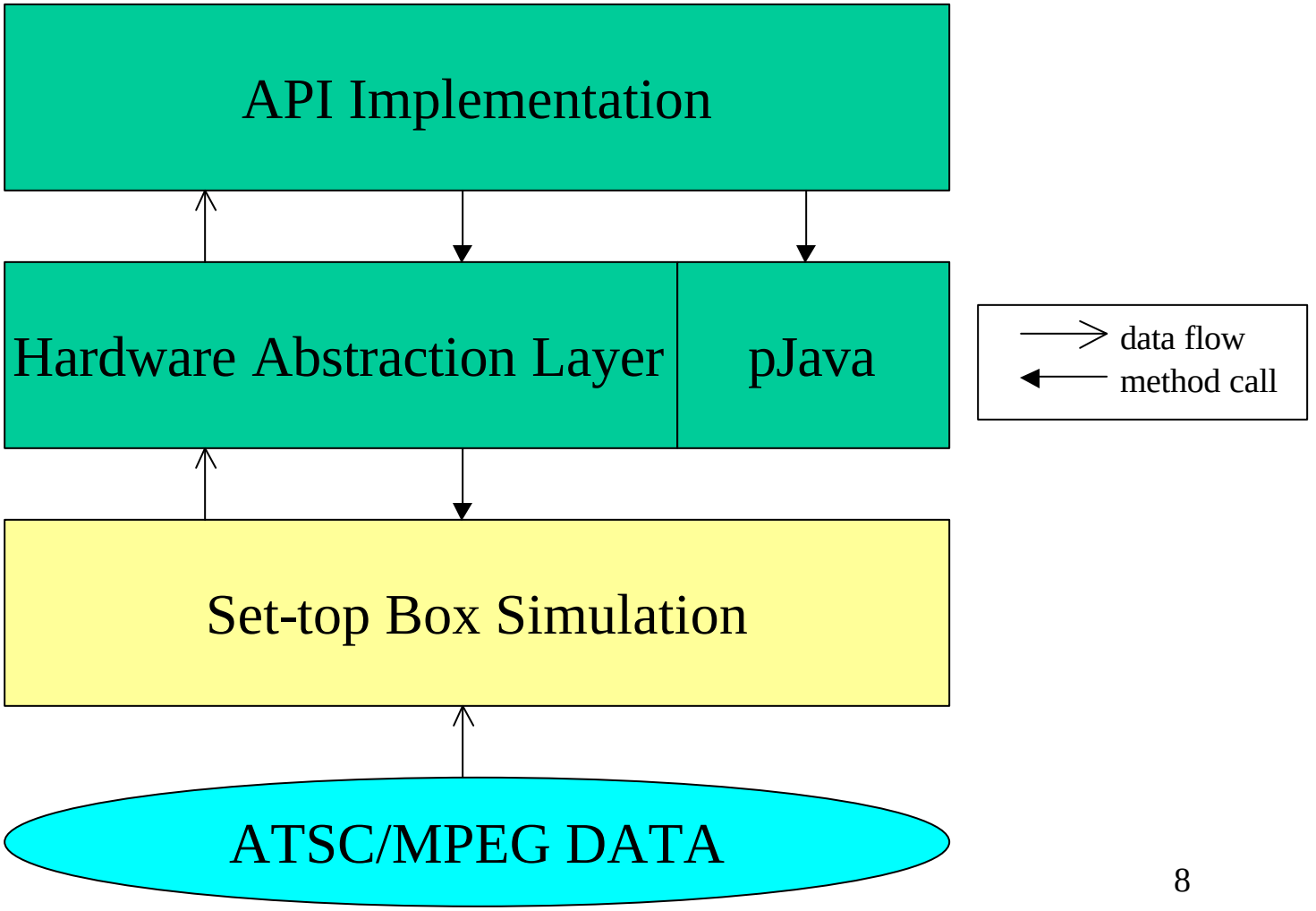
Reference Platform Scope (II)

- Set-top Box environment
  - Simulation
    - Java simulation
  - Real-time Emulation
    - Targeted to begin in the Fall/2000
  - Commercial Receivers
    - Will work with manufacturers in a collaborative effort

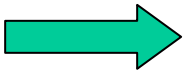
API Implementation Design Goals

- Portable, Semantic Clarity
  - Java implementation
  - Intermediate software layer between API implementation and STB environment
  - no native code
  - initially low priority given to performance and system constraint issues

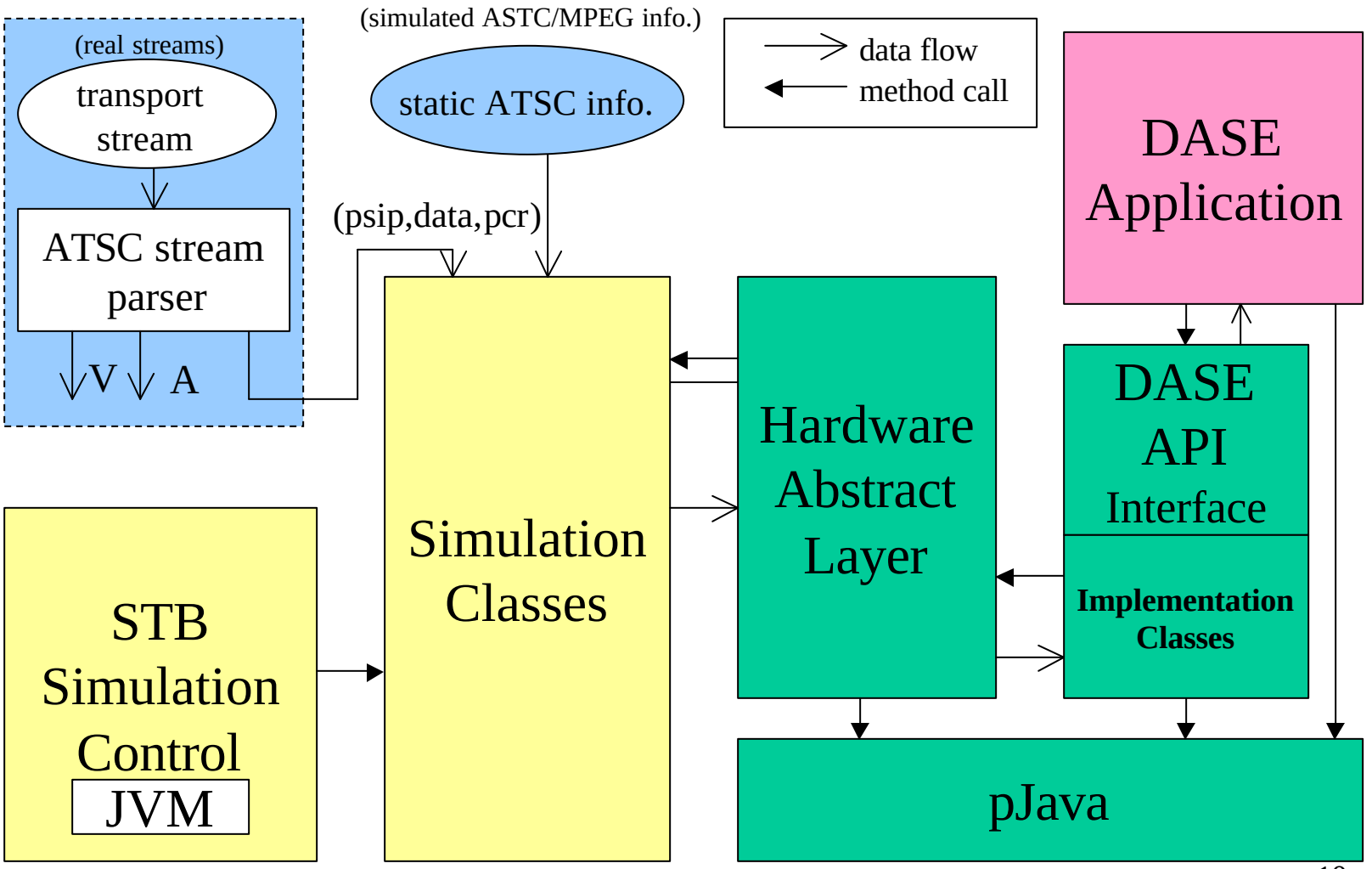
Reference Platform Design Stack



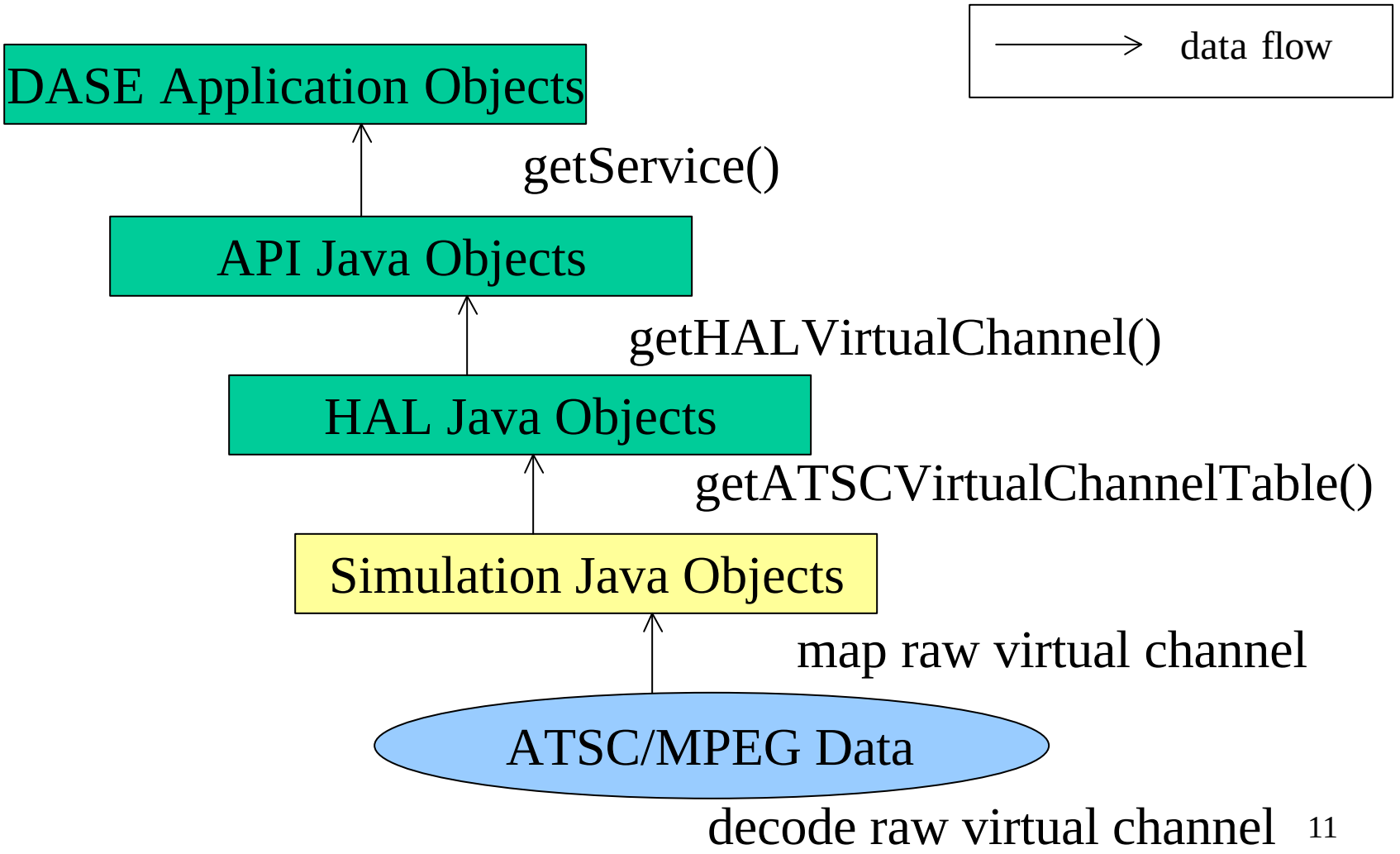
Hardware Abstract Layer

- intermediate software layer between API implementation and STB environment
- portable layer
- common interface that abstracts lower layer
- transform low level data to high level objects
  - ATSC table data  high-level objects
  - *merges* ATSC tables

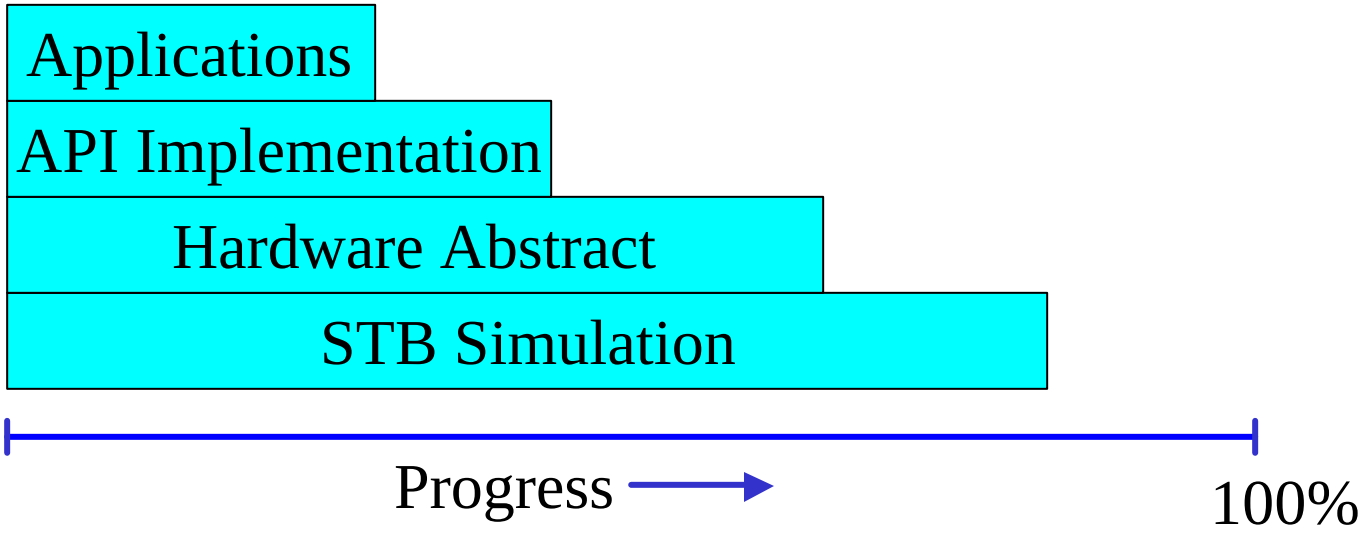
Reference Platform Architecture



Data Flow Example



Current Status



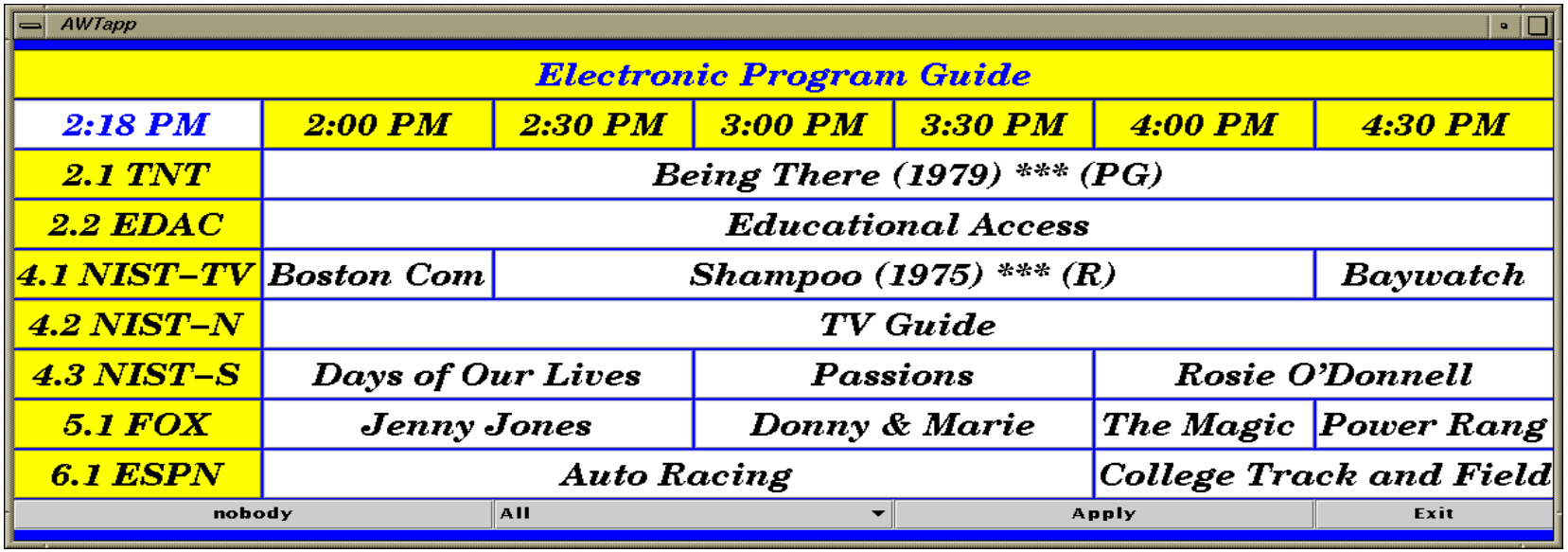
- API Packages
  - Service, Navigation, Guide
  - Locator, SI
  - Application, X-let
  - User, Registry, Preferences
  - Media, Graphics

Sample Applications

- Native EPG
  - SI Database access
  - User's Preferences
- Downloadable X-lets
  - SI Database access
  - Application Management
  - Media
- Demonstrations at reception tonight

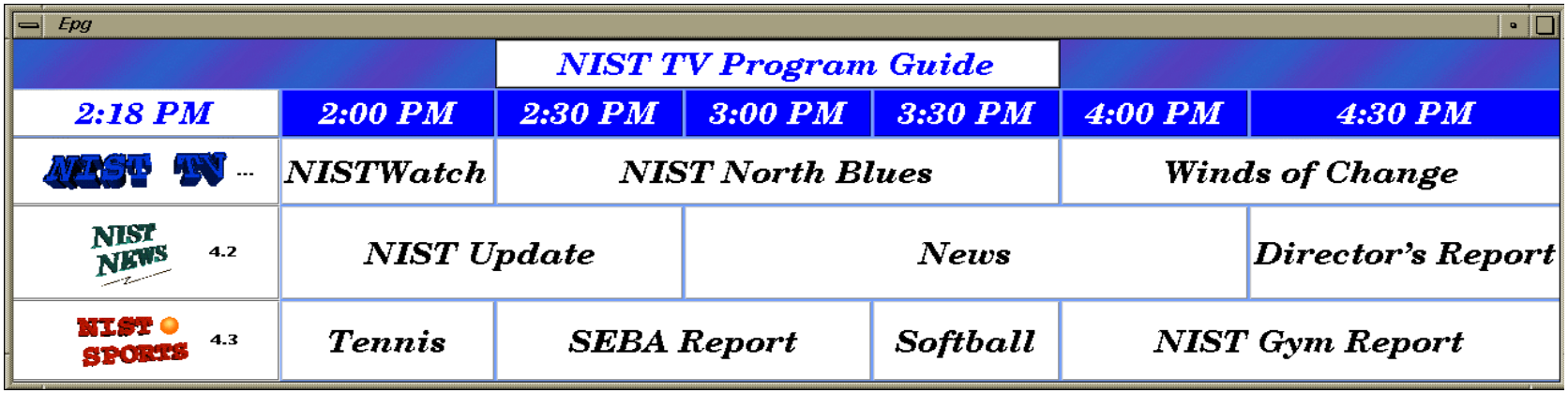
Electronic Program Guide

- native EPG application
- SI Database APIs
- user preferences APIs



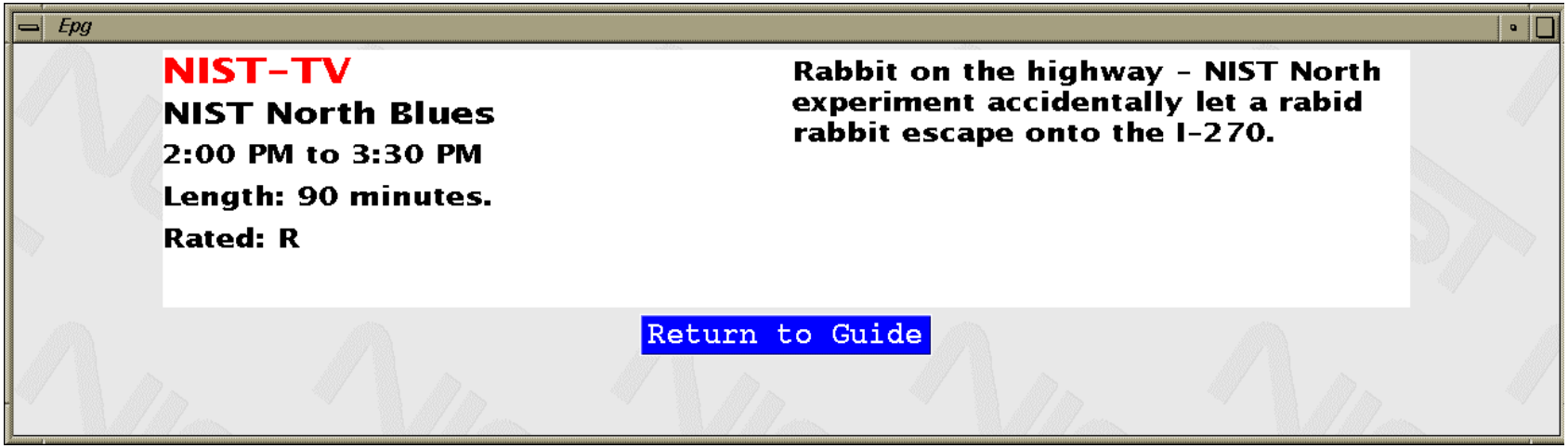
EPG X-let

- downloadable provider specific EPG
- auto-start application, shared
- EPG appears on-screen if present in bitstream
- SI Database APIs (VCT, EIT)



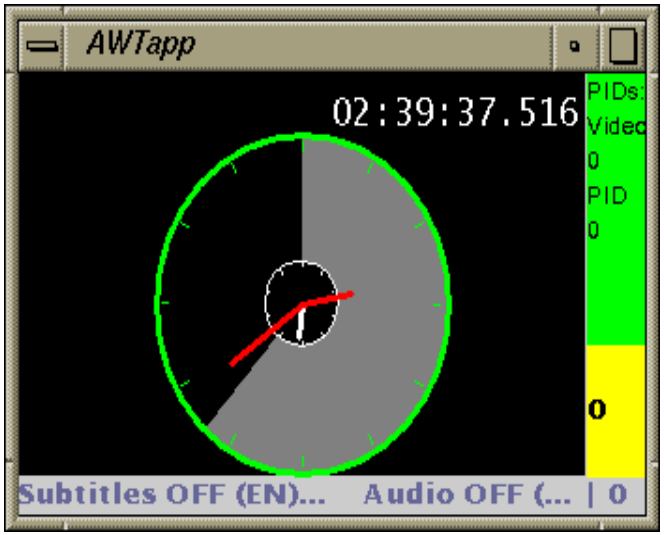
EPG X-let

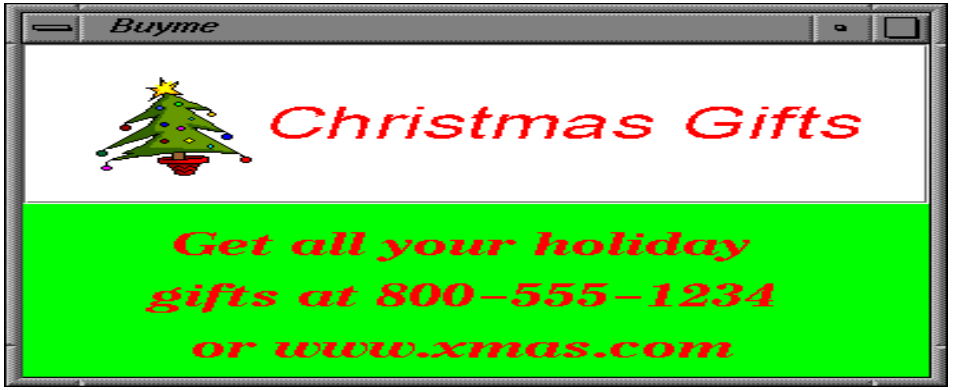
- additional ATSC/MPEG tables provide more in-depth information on programs
- SI Database APIs (VCT, EIT, ETT, RRT)
- asynchronous data retrieval



Buyme X-let

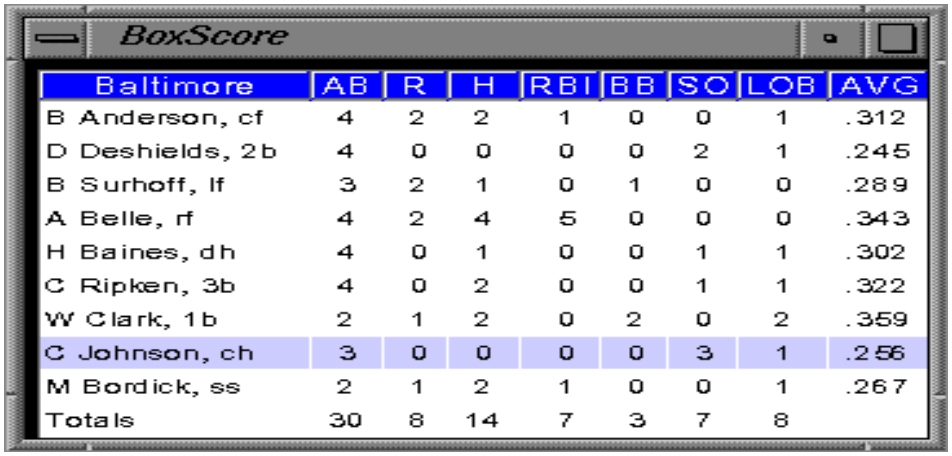
- implements X-let and MediaTimeEventListener interfaces
- auto-start X-let injected into simulation
- Application Manager detects the X-let and starts it
- registers to receive media time event
- X-let appears at requested time





Baseball Statistics X-let

- user-start application
- viewer activates X-let, Application Manager starts it
- statistics appears as long as X-let is present in the bitstream
- X-let disappears when no longer present in bitstream or stopped by the viewer



Going Forward

- Complete prototype API implementation
  - adjust to changes in specification
  - build lower-level infrastructure
- possibly expand role to other components
- Port to other STB environments
  - real-time emulation
  - commercial receivers

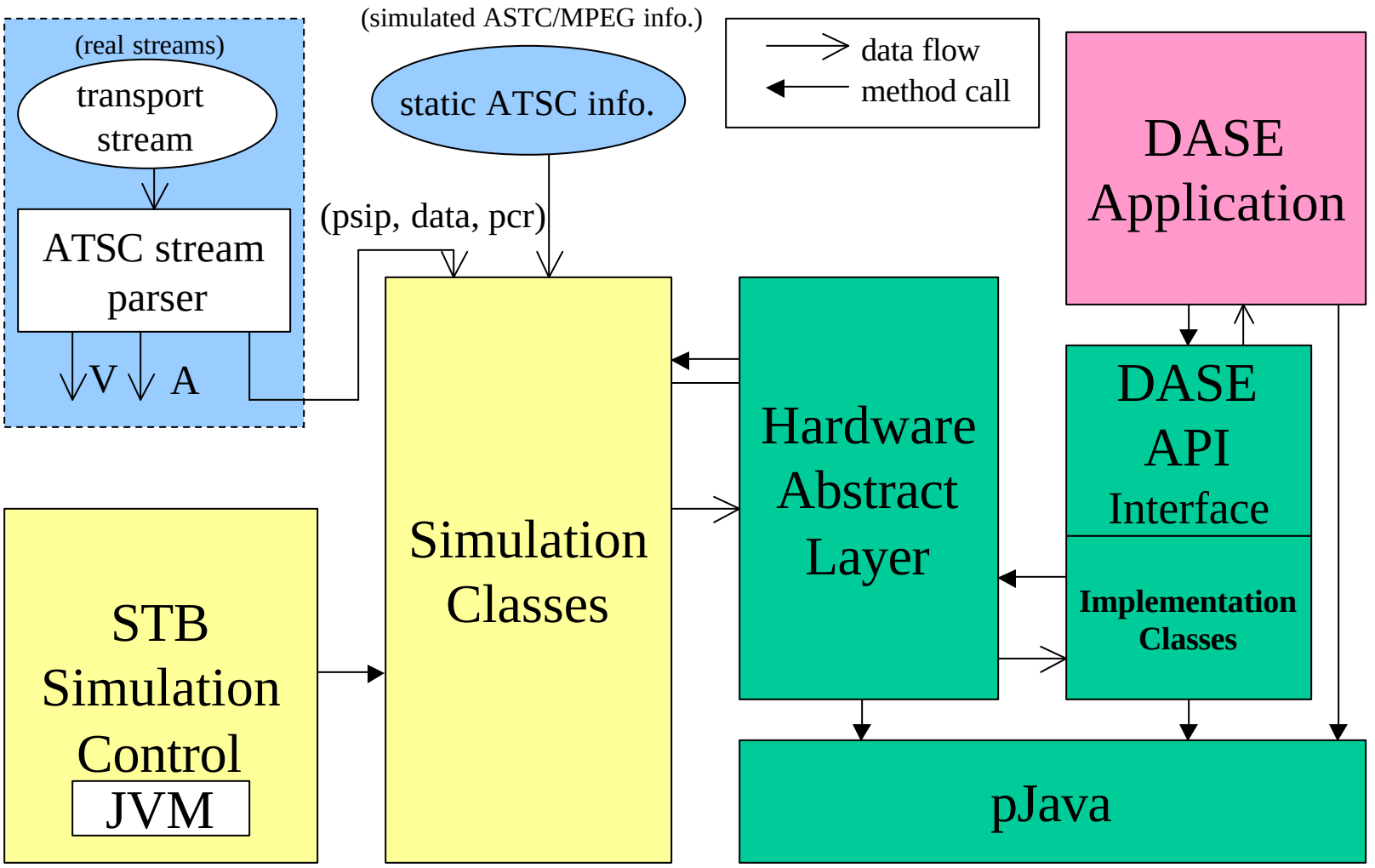
NIST Reference Platform Resources

- Reference Implementation Source Code
- STB Simulation Platform Source Code
- Sample Applications
- Unit Tests
- Tools
- JavaDoc
- Documentation

Availability

- Free and available to anyone
- www.dase.nist.gov

Reference Platform Architecture



NIST Settop Box Simulation

Wayne Salamon
<wsalamon@nist.gov>

May 23, 2000

Overview

- The need for the STB simulation
- Simulation design
- Simulation components
- Data flow from simulation to API
- Java class code reuse
- Executing native applications and Xlets

The Need for an STB Simulation

- DASE API retrieves info from underlying system
- Simulate successful as well as error scenarios
- Maintain state of users and preferences across API test runs
- Used to test Xlets

Simulation Design

- Independent of the API and HWAbstract classes
- Implementation based on API requirements
- Performs data management, not information management
  - For example, applies no semantic meaning to the contents of the PSIP tables

Simulation Design (cont.)

- Maintains the table consistency and will not return a partially completed table
- Extracts modules from the Data Carousel but doesn’t interpret the data
  - For example, Xlet classes and data are maintained as arrays of bytes

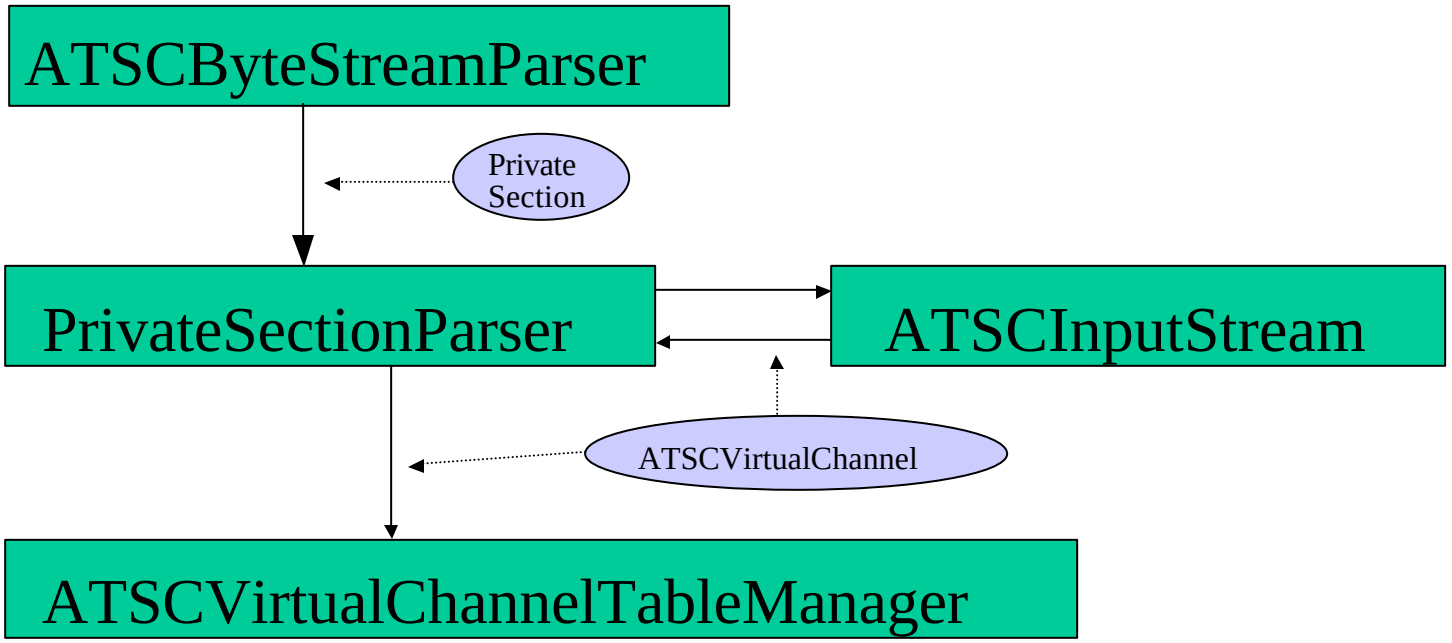
Simulation Components

- Small C program for initialization, remainder in Java
- ATSC and MPEG table processing and management
- Data carousel module processing
- Other data managers: user, preferences, etc.
- STB state: Power status, resource availability, etc.

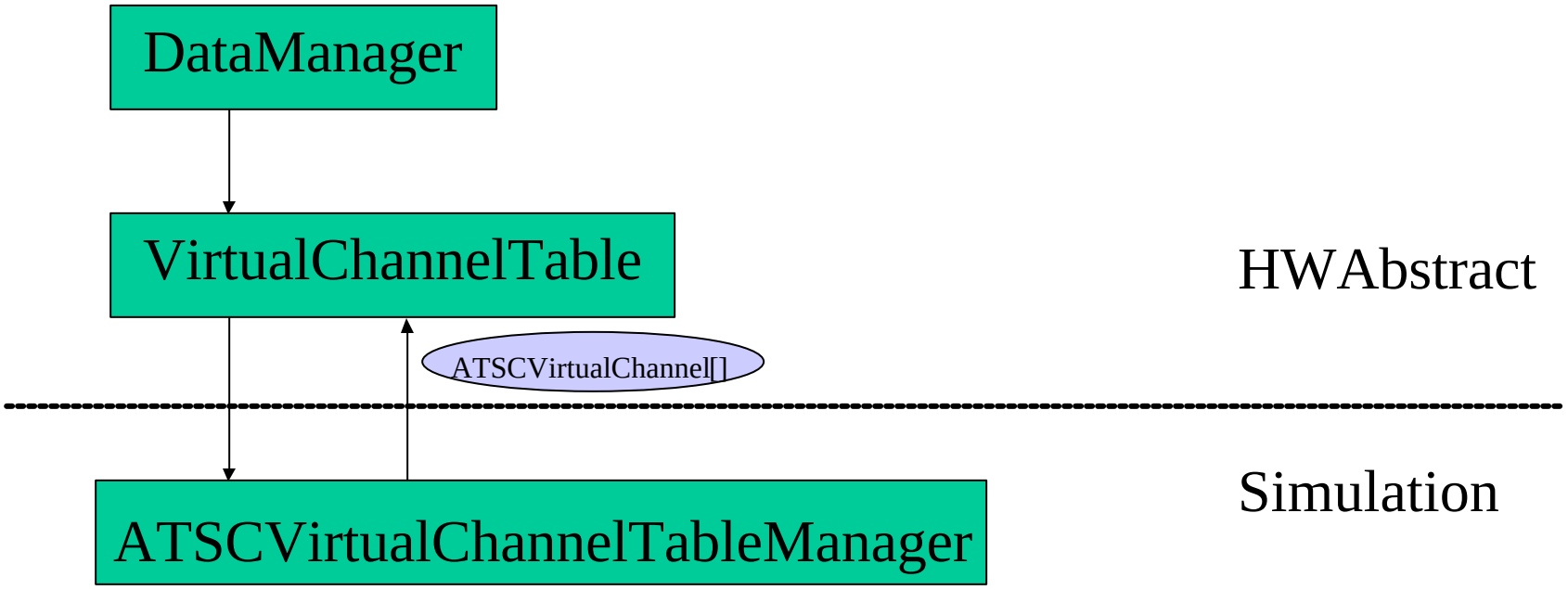
Data Flow Example

- Example of how the Virtual Channel Table is extracted by the simulation and presented to the HWAbstract layer
- Trace the class interactions needed to provide this data
- All other tables are handled in a similar manner

Extracting Virtual Channels



HWAbstract Channel Retrieval



Reuse of the Java classes

- Parsing is separated from table management
- The Huffman decoding is done by a separate utility class which is used by the parser classes
- Carousel module processing is an important function and the classes in the simulation can be reused for S13 protocol handling

Executing a Native Application

- Native applications have access to the DASE API and public methods of the HWAbstract classes
- Can be executed from the simulation command line or another native application
- Classes must be found in the CLASSPATH or the STBSIMCLASSPATH

Running an Xlet

- Xlets can be executed from the simulation command line or injected through the data stream
- Injected Xlets are either auto-started or user-started
- Xlets can be controlled with the Application Selection program

Conclusion

- STB Simulation is not dependent on the API implementation
- Forms the basis of a Xlet development environment
- Portable: Runs on multiple platforms
- Many of the STB classes can be reused
- Will be included in the NIST Reference Platform distributed via the Web site *www.dase.nist.gov*